# Trabalho Prático

CI316 – Programação Paralela
Prof. Daniel Weingaertner

Equações Diferencias Parciais
Método de Jacobi
Método de Gaus-Seidel
Red-Black Gaus-Seidel
Trabalho Prático

# Equações Diferenciais Parciais

- Equações Diferenciais Parciais (PDE) são equações diferenciais que contém ***funções multivariadas*** desconhecidas e suas derivadas parciais.
- PDEs são utilizadas para descrever diversos fenômenos como: som, calor, eletrostática, fluxo de fluidos, elasticidade e mecânica quântica.
- Métodos numéricos para solução: elementos finitos; diferenças finitas e volumes finitos

# Discretizando PDEs

- Equação Diferencial Parcial:

$$-\Delta u(x,y) + k^2 u(x,y) = f(x,y)$$

$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2}\right) + k^2 u(x,y) = f(x,y)$$

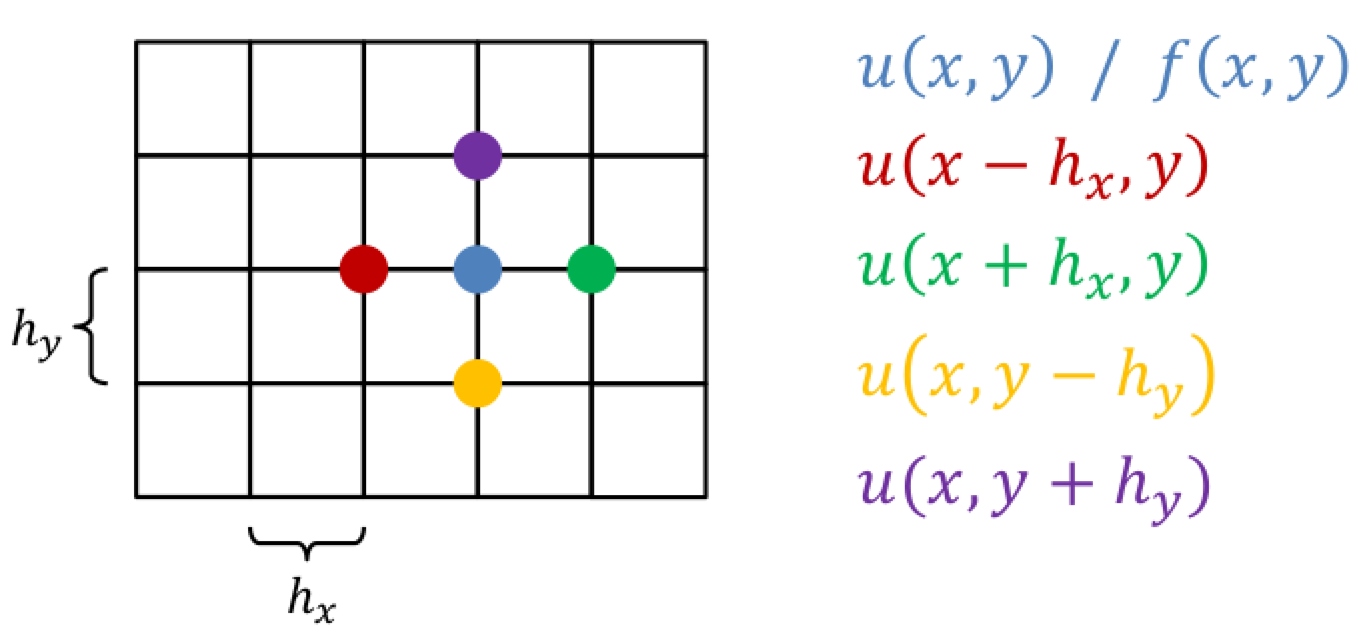
- Discretização usando o quociente diferencial:

$$-\left(\frac{u(x-h_x,y) - 2u(x,y) + u(x+h_x,y)}{h_x^2} + \frac{u(x,y-h_y) - 2u(x,y) + u(x,y+h_y)}{h_y^2}\right) + k^2 u(x,y) = f(x,y)$$

# Discretizando PDEs

$$-\frac{1}{h_x^2}\left[u(x-h_x,y)+u(x+h_x,y)\right]-\frac{1}{h_y^2}\left[u(x,y-h_y)+u(x,y+h_y)\right]$$
$$+\left(\frac{2}{h_x^2}+\frac{2}{h_y^2}+k^2\right)u(x,y)=f(x,y)$$

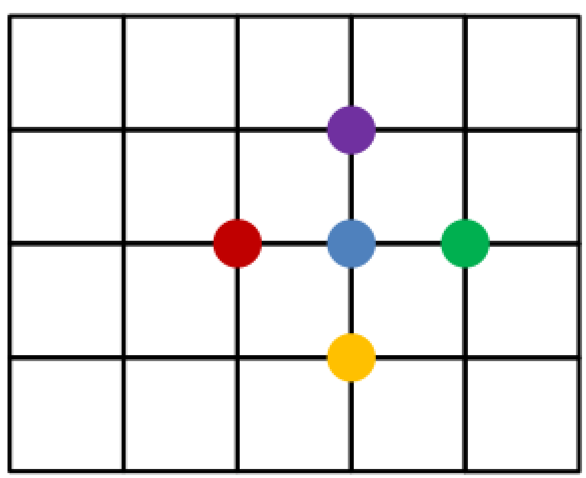
- A PDE é resolvida num **domínio discreto** $\Omega$



# Discretizando PDEs

- Para cada ponto $u_{x,y}$ formulamos uma eq. linear:

$$-\frac{1}{h_x^2}\left[u_{x-1,y}+u_{x+1,y}\right]-\frac{1}{h_y^2}\left[u_{x,y-1}+u_{x,y+1}\right]+\left(\frac{2}{h_x^2}+\frac{2}{h_y^2}+k^2\right)u_{x,y}=f_{x,y}$$




- Formulando esta eq. para cada ponto da grade temos um Sistema Linear (SL): $A\vec{x}=\vec{b}$

# Método de Jacobi

- O Método de Jacobi resolve o SL **iterativamente** de acordo com a fórmula:

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij} x_j^k\right)$$

  - Resolve a i-ésima equação utilizando os valores da iteração anterior;
  - Como não há **dependência de dados**, cada $x_i^{k+1}$ pode ser calculado paralelamente;
  - Convergência é mais lenta;
  - Utiliza duas grades: iteração atual e anterior.

# Método de Gauss-Seidel

- Resolvendo a i-ésima equação:

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \underbrace{\sum_{j<i} a_{ij}x_j^{k+1}}_{\text{same iteration}} - \underbrace{\sum_{j>i} a_{ij}x_j^{k}}_{\text{previous iteration}}\right)$$

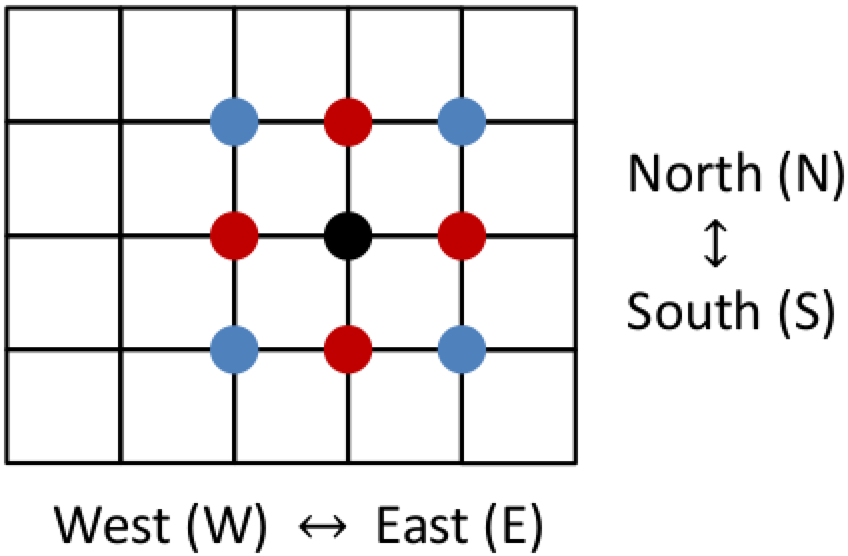
  - Algumas variáveis vem da iteração anterior, algumas da iteração atual;
  - Apresenta melhor convergência;
  - Possui **dependência de dados**;
  - Utiliza apenas **uma** grade: atualização *in place*.

# Red-Black Gauss-Seidel

- Dependências de dados do ponto central:

$$u_{x,y} = \frac{1}{\left(\frac{2}{h_x^2}+\frac{2}{h_y^2}+k^2\right)}\left(f_{x,y} + \frac{1}{h_x^2}\left[u_{x-1,y} + u_{x+1,y}\right] + \frac{1}{h_y^2}\left[u_{x,y-1} + u_{x,y+1}\right]\right)$$

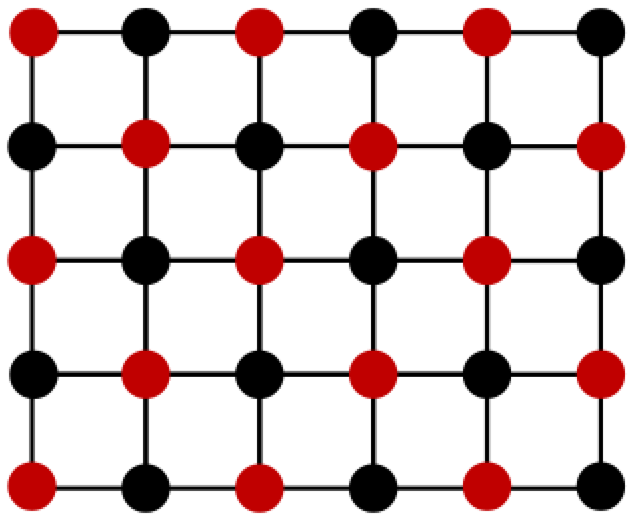
$$\begin{bmatrix} & -\frac{1}{h_y^2} & \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2}+k^2 & -\frac{1}{h_x^2} \\ & -\frac{1}{h_y^2} & \end{bmatrix}$$




  - <u>Não há dependências</u> em **NW**, **NE**, **SW** e **SE**
  - <u>Há dependências</u> em **W**, **E**, **N** e **S**

# Red-Black Gauss-Seidel

- Paralelização: ***checkerboard reordering***



$$\begin{bmatrix} & NORTH & \\ WEST & CENTER & EAST \\ & SOUTH & \end{bmatrix}$$

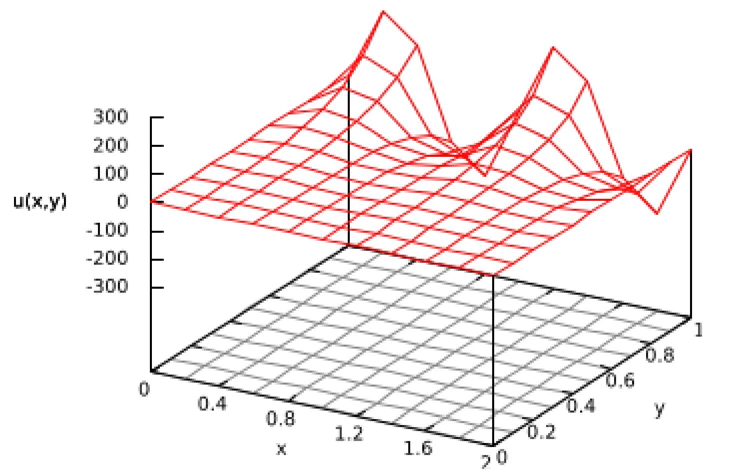
A general stencil for the PDE

  - **Ideia:** dividir as variáveis em dois grupos (***red***, ***black***), de forma que não há dependências entre os grupos;
  - **Consequência:** atualização dos grupos pode ocorrer de forma independente, i.e., paralela.

# Resolvendo PDEs

- Para resolver a PDE $-\Delta u(x,y)+k^2 u(x,y)=f(x,y)$ não é necessário construir a matriz **A**;
- Apenas precisamos conhecer o **stencil**:

$$\begin{bmatrix} & -\frac{1}{h_y^2} & \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2}+k^2 & -\frac{1}{h_x^2} \\ & -\frac{1}{h_y^2} & \end{bmatrix}$$




A solution of the PDE

# Resolvendo PDEs

- Cada ponto $u_{x,y}$ da grade é calculado da seguinte maneira (a cada iteração):

$$u_{x,y} = \frac{1}{\left(\frac{2}{h_x^2}+\frac{2}{h_y^2}+k^2\right)}\left(f_{x,y} + \frac{1}{h_x^2}\left[u_{x-1,y} + u_{x+1,y}\right] + \frac{1}{h_y^2}\left[u_{x,y-1} + u_{x,y+1}\right]\right)$$

- O **stencil** equivale à **regra de atualização**;

# Aumentando Eficiência da Cache

- Na atualização das variáveis ***red***, apenas os valores de $f(x, y)$ referentes a estas variáveis são lidos;
  - Dividindo os valores de $f(x, y)$ em dois conjuntos aumenta a **localidade espacial**;
- Enquanto as variáveis ***red*** são apenas escritas, as variáveis ***black*** são apenas lidas (e v.v.);
  - Processadores possuem instruções para escrita di-reta, sem passar pela cache: **non-temporal writes**
  - Separação de ***red*** e ***black*** permite **non-temporal writes**

# Acesso à Memória Compartilhada

- Memória é atribuída ao processador que primeiro escreve nela: ***first touch***;
  - Para evitar acessos não locais, inicialize os dados com a ***thread*** que irá processar os dados.
  - Evite migração de ***threads*** para outros processadores "pinando-as" (*pinning*).
    - LikwidPin http://code.google.com/p/likwid/wiki/LikwidPin)
    - GCC OpenMP http://gcc.gnu.org/onlinedocs/libgomp.pdf)

# Trabalho Prático

Entrega até **25.maio.2014, 23h59m59s**

# Trabalho

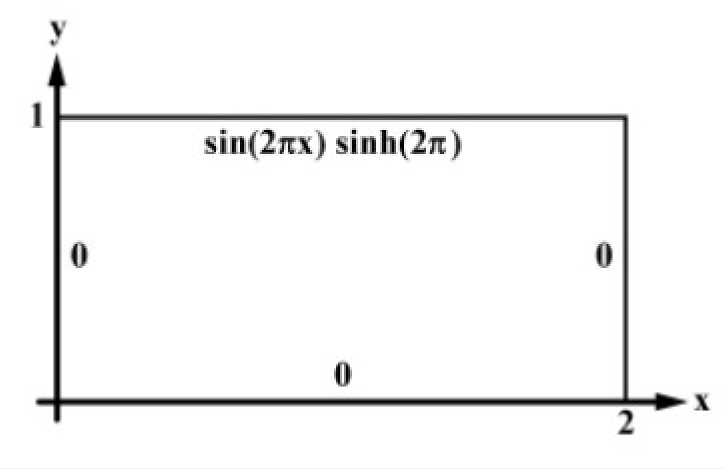
- O trabalho consiste em calcular a solução discreta para a seguinte PDE:

$$-\Delta u(x,y)+k^2 u(x,y)=f(x,y), \quad (x,y)\in\Omega$$

onde $k=2\pi$ e o lado direito da equação é definido por $f(x,y)=4\pi^2\sin(2\pi x)\sinh(2\pi y)$. O domínio é definido por $\Omega=[0,2]\times[0,1]$. Nas fronteiras $\partial\Omega$ use:

$$u(x,1)=\sin(2\pi x)\sinh(2\pi)$$

$$u(x,0)=u(0,y)=u(2,y)=0$$



# Trabalho

- Para resolver a PDE utilizar uma discretização em Elementos Finitos:
  - O tamanho da malha da grade $h_x$ e $h_y$ deve ser escolhido de acordo com os parâmetros $n_x$ e $n_y$ :

$$h_x = \frac{2}{n_x} \qquad h_y = \frac{1}{n_y}$$

  - Escolha uma estrutura de dados eficiente para representar o Sistema Linear resultante;
  - Use **zero** como estimativa inicial para a solução;
  - Escolha um layout eficiente para as variáveis e termos independentes do seu sistema;

# Trabalho

- Utilize os seguintes métodos para resolver o SL:
  - Jacobi
  - Red-Black Gauss-Seidel
- Execute um número fixo **c** (definido na linha de comando) de iterações do método;
- Meça o tempo de execução (*wall clock time*);
- Calcule a norma L2 do resíduo: $\|r\|_2 = \|\vec{b} - A\vec{x}\|_2$

# Trabalho

- Imprima o tempo e o resíduo em **stdout**;
- Imprima a solução computada em um arquivo chamado “solution.txt”, em um formato que pode ser plotado pelo programa **gnuplot**.
- Utilize C ou C++ e OpenMP;
- Utilize o compilador GCC;
- Gere relatórios de desempenho para 1, 2, 4, 8, 16 e 20 *threads* (não use hyper-threads).